

# Sequence Read Archive Overview

National Center for Biotechnology Information (NCBI)

National Library of Medicine

Version 1.0 Draft D May 26

## Table of Contents

An Introduction to the Sequence Read Archive .....	1
SRA Features .....	2
A Central Data Repository with Submission Flexibility .....	2
An Integrated Search and Analysis System .....	3
SRA Architecture .....	4
The Need for a New Paradigm in Massive Data Storage and Retrieval .....	4
SRA: A Dynamic Mix of Database and Local Computing .....	5
SRA Data Structure .....	7
The Necessity of Archival Data Storage .....	8
SRA Future Developments .....	9

## An Introduction to the Sequence Read Archive

The advent of massively parallel sequencing technologies has opened an extensive new vista of research possibilities — elucidation of the human microbiome, discovery of polymorphisms and mutations in individual genomes, mapping of protein–DNA interactions, and positioning of nucleosomes — to name just a few. In order to achieve these research goals, researchers must be able to effectively store, access, and manipulate the enormous volume of read data generated from massively parallel sequencing experiments.

In response to the research community’s call for such a resource, NCBI, EBI, and DDBJ, under the auspices of the International Nucleotide Sequence Database Collaboration (INSDC), have developed the Sequence Read Archive (SRA) data storage and retrieval system. The SRA not only provides a place where researchers can archive their sequence read data, but also enables them to quickly access known data and their associated experimental descriptions (metadata).

The SRA currently contains more than 7 terabases (7,000 gigabases) of sequence read data and is growing rapidly. Due to the regular exchange of data between NCBI, EBI and DDBJ — all of whom are using the same design model and code libraries in their respective sequence read archives — researchers will

be able to access the most up-to-date sequence read data from around the world.

## ***SRA Features***

### **A Central Data Repository with Submission Flexibility**

The SRA preserves all content submitted from the major sequencing technologies, and currently accepts submissions that originate in any of these major sequence read formats:

- SFF (Roche 454)
- Illumina Native
- Illumina SRF
- AB SOLiD Native
- AB SOLiD SRF
- fastq

Please note that additional formats will be supported as they become available.

SRA provides two venues for submissions: 1) An interactive [web-based interface](#) for occasional submissions, which requires only a brief registration prior to submission, and 2) An automated submission pipeline for centers making multiple submissions; this process uses XML to describe metadata and Sequence Read Format (SRF) as a common container file format.

SRA uses a high-speed file transfer protocol called fasp (Aspera, Inc., Emeryville, CA), which allows users to transfer files to and from the SRA at speeds up to 400 Mbps — many times faster than ftp. Details on obtaining the free client can be found in the [SRA Submission Guidelines](#).

The SRA's common, compact design allows for the storage and rapid retrieval of all types of data from massively parallel experiments, including:

- Reads and associated quality scores
- Trimming and other technical information
- Experiment metadata
- Secondary analyses typically performed on sequence read data, including:
  - Alignments
  - Small-scale assemblies
  - Oligo profiles
- Intensity data

Submission requirements for the SRA are flexible; a user can include all of the data elements listed above, or just a subset of them. For example, the user could submit descriptive information about a study or experiment once details

were decided, and submit the experimental results sometime in the future. The SRA submission process also allows users to modify their submissions and provides a Hold–Until–Published (HUP) option, where a user may delay release of the data until a specific date for the release of a publication.

Because the SRA's design allows for the submission of new (still under development) sequence read data analyses in "blob" form (virtually no internal structure), the SRA can be a "one-stop" submission resource for small projects, projects executed through automatic pipelines, and projects submitted by newer sequencing centers that have little experience interacting with NCBI.

The SRA also allows for the selective movement of older, less frequently used data element(s) to less expensive storage (tape or disc), or for the eventual discard of the data element(s) without having to reload any of the other data associated with these redistributed or discarded element(s).

For further information about submitting to SRA, please see the [SRA Submission Guidelines](#).

## **An Integrated Search and Analysis System**

The SRA's design allows users to quickly and precisely retrieve massively parallel experiment data of interest down to the level of individual reads. SRA data can be retrieved from the [SRA Download](#) page in either SRA Native format or as fastq formatted files.

### **Accessing SRA Data Using the Web Interface**

Since the SRA metadata is indexed in NCBI's Entrez search and retrieval system, users can access SRA content from the [NCBI home page](#), [PubMed](#), [the Genome Project database](#), [the Gene Expression Omnibus \(GEO\)](#), or [the Taxonomy database](#). Users can also search SRA through the use of the "Browse" and "Search" tabs on the [SRA home page](#). The SRA web interface allows the user to:

- Access any data type stored in the SRA independently of any other data type, (e.g., accessing read and quality data without the intensity data)
- Access reads and quality scores in parallel
- Access related data from other NCBI resources that are integrated with SRA
- Retrieve data based on ancillary information and/or sequence comparisons

- Retrieve alignments in “vertical slices” (showing underlying layered data) by reference sequence location [note: this capability currently is under development]
- Review the descriptions of studies and experiments (metadata) independently of experimental data

### **Accessing SRA Data using the System Development Kits**

The [SRA System Development Kits \(SDK\)](#) provide Application Programming Interfaces (APIs) that facilitate the accession and manipulation of larger quantities of data.

The “Read” SDK allows the user to programmatically access data housed within SRA and convert it from the SRA format to any of these major sequence read formats:

- AB SOLiD Native
- fastq
- SFF (Roche 454) [under development]
- Illumina Native

The “Read” SDK is designed to prevent accidental modification of the data, and is optimized to provide the user with the most efficient read possible.

The “Write” SDK, on the other hand, allows a user to read SRA data as well as convert (write) it from the major sequence read formats listed below into the SRA flexible format:

- fastq
- AB SOLiD-SRF
- AB SOLiD-Native
- Illumina SRF
- Illumina Native [under development]
- SFF [under development]

The “Write” SDK allows users to create a local archive using their own sequence read data, and in the future will be used for direct submissions to SRA.

## ***SRA Architecture***

### **The Need for a New Paradigm in Massive Data Storage and Retrieval**

NCBI began development of the SRA database in July, 2007 in response to the research community’s need for efficient and flexible ways to store and retrieve the large amounts of massively parallel sequencing data beginning to appear.

Now that the archive has reached an initial state of completion and is publically available at NCBI, it is being deployed at EBI (under the name European Read Archive, or ERA), and soon will also be deployed at DDBJ (under the name DDBJ Read Archive, or DRA). NCBI and EBI have already begun exchanging data, and once the DRA is in place at DDBJ, there will be a regular data exchange between all three INSDC members.

The initial design of the SRA was conceived by looking at the advantages and disadvantages inherent in relational databases and in file-based storage systems, and using the best aspects of each to create something new.

### ***Relational Databases Alone are Not the Answer***

Relational databases are good for recording and manipulating related data: they can index, make complex arbitrary joins, and process complex queries. Relational databases, however, are not a practical approach for the long-term storage of the terabyte and petabyte amounts of data generated from massively parallel sequencing experiments: they are bulky, require significant management, and are inflexible and costly in terms of storage space.

### ***File-based Storage Systems Alone are Not the Answer***

Simple file-based storage systems have many advantages to recommend them for storing large amounts of data: they are lightweight; they make use of the file and directory systems already available; they store data according to an object model (i.e. a run is in a single file, which is an object that can be accessed, shipped, modified, removed, etc.). However, file-based storage systems lack support for indexed queries and cannot create necessary relationships. Archiving data in a file-based storage system often means using the tar (tape archive) format and applying a compression utility like gzip or bzip2 to produce a compressed tar file, making the data difficult (or impossible) to access in a repository setting.

### ***A Hybrid Storage and Retrieval System***

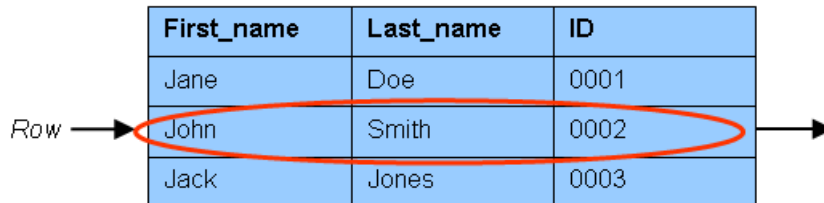
In order to store and retrieve the enormous amount of data generated by massively parallel sequencing technologies, NCBI, EBI and DDBJ needed to create a data repository that has much of the power of a relational database while being lightweight, transportable and flexible like flat-file storage. The solution was to create a hybrid relational database with a file-based and column-oriented design.

## **SRA: A Dynamic Mix of Database and Local Computing**

The SRA's design — a novel combination of a file-based, column-oriented design and a relational database — offers great versatility in storage and management of data. Search and retrieval services can be based not only at NCBI but also locally once users move their data into the SRA flexible format.

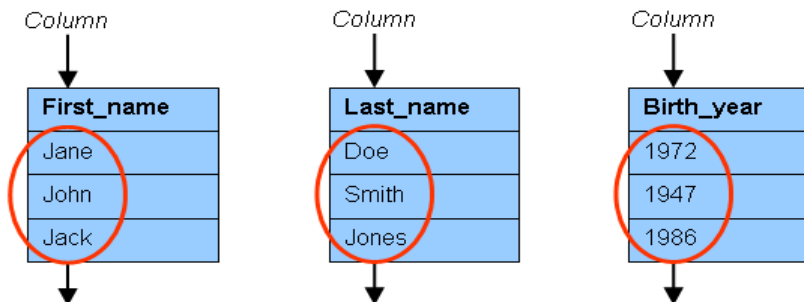
### ***Row vs. Column-Oriented Database Design***

Row-oriented databases store data as a series of row structures, where each structure contains one or more fields (unique data types arranged in columns) linked together in a table. In the row-oriented system, a user approaches the data from left to right in a single row.



First_name	Last_name	ID
Jane	Doe	0001
John	Smith	0002
Jack	Jones	0003

Column-oriented databases turn this structure on its side, so to speak, and store the data as columns, where each data type is stored as a series independently in its own column (still associated with its unique ID). In the column-oriented design, a user approaches each data type as an independent series moving from the top of the column downward to the bottom of the column.



First_name
Jane
John
Jack

Last_name
Doe
Smith
Jones

Birth_year
1972
1947
1986

The advantage of row-oriented databases is that they link together all the fields (data types) in a single row so that the entire row can be retrieved with a single read. This becomes a disadvantage, however, when applied to the problem of massive data storage:

- Since multiple data types are involved, compression and/or packing of data in row-oriented tables is difficult and results in inefficient use of storage space.
- Because the fields are linked together in each row, the removal of one field necessitates the re-write of the entire data table, making the addition or removal of specific fields difficult.

Column-oriented databases, in contrast, are able to achieve improved storage because there is only one data type per column, and also have greater retrieval efficiency. In addition, if a column-oriented database is properly designed, a column (data type) can be added or removed independently of the other columns

in the table, so there is no need to re-write an entire data table for every addition or deletion of a data type.

### ***The SRA Hybrid Design***

The SRA's file-based, column-oriented design makes use of the file system to keep the data columns physically separate. Each data column within the SRA design model is packaged in its own UNIX file rather than in a database. This makes it possible to store the most frequently accessed data series (e.g., "fastq" data) in fast, near storage, and less frequently accessed data (e.g., intensities and reads) in slower, bulk storage, such as tape or disk, located at the repository where the data were submitted (NCBI, EBI, or DDBJ). Users can obtain whichever data columns/files are desired. The design allows users to access and read any SRA sequence read data in random order, stream it quickly, or get reads and quality scores in parallel. The NCBI relational database portion of the SRA hybrid design model serves to track runs and components, while the SRA toolkit operates using directories and files, so it can easily bring the power of the SRA approach to a local computer.

### ***SRA Data Structure***

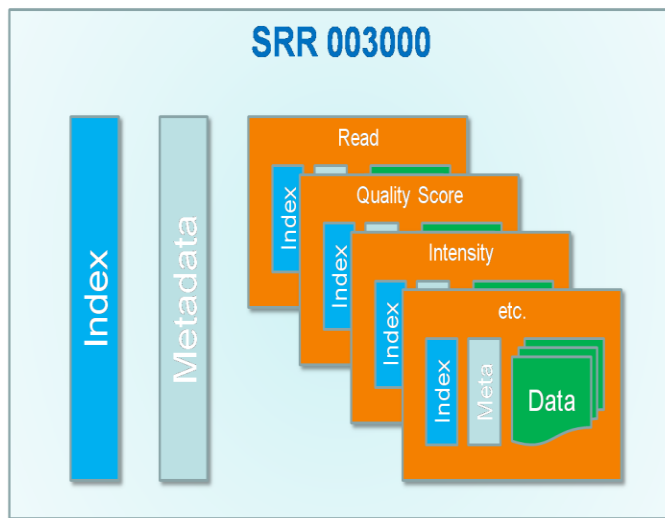
As mentioned above, SRA uses file-based data management, where the base unit is a column packaged within its own UNIX file. Each column represents a single data type (read, quality score, or intensity data, etc.), and the UNIX file holding the column contains not only the particular data type, but also the index of identifiers for each member of that data type and a minimal description (SRA descriptor) of each:



If, for example, the column represented above holds read data for a single run within an experiment, then this “read column” (file) would contain the series of template reads (data) generated in the run, the identifiers for each read (which are given serial ID numbers based on the run ID to save space), and a small amount of SRA descriptor information that describes the read (name [alias] and plate location).

The “columns” (files) are then organized together into a “run” — a “table” in database parlance. The “run” (table) groups columns that contain the data gathered for a sample or sample bundle in a particular experiment into a single

structure (the number of columns in a run is arbitrary and depends on the number of data types available):



In the example above, the run accession number is SRR003000; this particular run contains a series of Read data, Quality Score data, Intensity data, as well as other data types that may exist (“etc.”), where each data type is contained within its own column (file). In addition, the run “table” contains a substantial amount of SRA descriptor information, including technical information about the instrument model, date of run, run center, plate statistics, brief experimental description, etc., as well as an overarching index for the data housed within the run.

A unique feature of this type of table is that it takes its runtime structural definition from the contents of the file system — that is, it determines its component columns dynamically when it is opened. This feature provides the user with great versatility when it comes to archiving data since old data types can easily be removed and replaced with new data types. For example, if new quality data becomes available, the old quality data column can be removed and replaced with a new quality column, and even if the new quality data is of an entirely different type than the old column, the table will resolve. Similarly, if a user no longer wished to archive intensity data, the intensity column could easily be removed and the table would still resolve upon opening.

### ***The Necessity of Archival Data Storage***

There is a great deal of discussion in the community involved with massively parallel sequencing regarding the necessity of archiving the various data types generated from sequence read sequencing experiments:

#### ***Intensity Data***

Some within the community feel that intensities are no longer needed once base calls are made, while others believe they should be archived because bases may



need to be re-called — for example, if new and improved base-calling algorithms are developed. SRA, therefore, was designed to enable archiving of intensity data. Many project leaders are choosing to archive intensities for early experiments completed prior to the establishment of optimal base calling, or for important projects where it may be difficult to re-sequence the samples. It may be sufficient, however, to deposit only the reads for projects such as ChIP-Seq experiments. For those projects depositing intensity data, the SRA provides the option to discard the intensity data at a later date if the community deems that it is no longer necessary.

### ***Read Data***

The SRA also was designed to archive read data because the data can be used in a variety of important ways:

- For alignments when improved alignment algorithms become available
- To regenerate an alignment when a reference assembly is updated
- To generate an alignment to another reference assembly (e.g., European, Asian, or African reference human genome assemblies)
- To pool data across different experiments or create experimental sub-sets from within an experiment

### ***Alignment Data***

The SRA is developing the capability to archive alignment data, as this data may be used:

- For re-analysis for purposes different from those intended in the original experiment (e.g., alignment data from a gene expression experiment could be used for SNP verification)
- To verify simple summaries like histograms
- To generate SNP calls, CNV calls, or different histograms

### ***SRA Future Developments***

In addition to developing an archival storage system for sequence read alignment and assembly records, SRA will continue to support new platforms and sequencing technologies as they become available.